Mathematical Biosciences and Engineering, 16(3): 1190–1209.
DOI: 10.3934/mbe.2019057
Received: 20 November 2018
Accepted: 14 January 2019
Published: 18 February 2019

http://www.aimspress.com/journal/MBE

Research article

A modified comprehensive learning particle swarm optimizer and its application in cylindricity error evaluation problem

Qing Wu[1], Chunjiang Zhang[2,*], Mengya Zhang[1], Fajun Yang[3] and Liang Gao[2]

[1] College of Engineering, Huazhong Agricultural University, Wuhan, Hubei, China, 430070
[2] State Key Lab of Digital Manufacturing Equipment & Technology, Huazhong University of Science and Technology, Wuhan, Hubei, China, 430074
[3] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798

* **Correspondence:** Email: zhchj1989@gmail.com; Tel: +8618372099229.

Abstract: Particle swarm optimizer was proposed in 1995, and since then, it has become an extremely popular swarm intelligent algorithm with widespread applications. Many modified versions of it have been developed, in which, comprehensive learning particle swarm optimizer is a very powerful one. In order to enhance its performance further, a local search based on Latin hypercube sampling is combined with it in this work. Due to that a hypercube should become smaller and smaller for better local search ability during the search process, a control method is designed to set the size of the hypercube. Via numerical experiments, it can be observed that the comprehensive learning particle swarm optimizer with the local search based on Latin hypercube sampling has a strong ability on both global and local search. The hybrid algorithm is applied in cylindricity error evaluation problem and it outperforms several other algorithms.

Keywords: comprehensive learning particle swarm optimizer; Latin hypercube sampling; cylindricity error evaluation

1. Introduction

Global optimization problems widely exist in engineering application and scientific research. For linear and convex function optimization problems, the theories and corresponding optimization

algorithms are well-developed. However, these algorithms often fail to solve non-convex and non-differentiable problem with many optima. Since 1980s, intelligent optimization algorithms or so-called mate-heuristic algorithms have been developed. They can obtain near-optimal solutions even optima in reasonable time. Classic meta-heuristic algorithms include Simulated Annealing (SA) [1], Genetic Algorithm (GA) [2], Particle swarm optimizer (PSO) [3], Differential Evolution (DE) [4,5], Ant Colony Optimization (ACO) [6], Electromagnetism-like Mechanism Algorithm 7 etc. They have been used in extensive applications in areas such as engineering optimization [8–10] and scheduling optimization [11–13] etc.

Particle swarm optimizer (PSO) was first proposed by Eberhart and Kennedy in 1995 3, and since then, it has become a very popular algorithm since PSO shows many advantages such as simple implementation, good optimization capability and minimum mathematical processing. In the review paper [14] about nature-inspired intelligence, PSO was considered one of the most popular nature-inspired algorithms according to related publications. However, the original PSO algorithm may get trapped easily in local optima for multimodal optimization problems. In order to improve the search ability of PSO, an increasing number of modified PSO versions have been proposed, including PSO with inertia weight (PSO-*w*) [15], PSO with constriction factor (PSO-cf) [16], fully informed particle swarm (FIPS) [17], unified particle swarm optimization (UPSO) [18], comprehensive learning particle swarm optimizer (CLPSO) [19], cellular particle swarm optimization (CPSO) [20], and adaptive division of labor PSO (ADOLPSO) [21] etc.

Among these modified versions of PSO, the comprehensive learning particle swarm optimizer (CLPSO) proposed by Liang et al. 19 is a quite famous one which shows a superior ability for solving multimodal problems. Since CLPSO was published in 2006, it has been cited for 2500+ times in google scholar until now. Some researchers have developed new algorithms based on CLPSO. Peng and Lu [22] proposed a hierarchical PSO based on CLSPO. Gülcü and Kodaz [23] proposed a better variation of CLPSO which is named parallel comprehensive learning particle swarm optimizer (PCLPSO). These studies show that CLPSO has a strong vitality in the area of metaheuristic algorithms but there is still room for improvement.

Adding other local search methods to PSO is a very popular way to improve its performance. Bao, Hu and Xiong [24] combined PSO with pattern search for SVMs parameters optimization. Petalas, Parsopoulos and Vrahatis [25] used the random walk with directional exploitation local search in PSO. Jia et al. [26] combined the canonical PSO with a chaotic and Gaussian local search procedure. The cellular particle swarm optimization (CPSO) [20] essentially used a local search based on cellular automata. Wu et al. [27] proposed a superior solution guided particle swarm optimization (SSG-PSO) combined with four gradient-based or derivative-free local search methods. In these algorithms, the local search usually focuses on enhancing the ability of intensification. Is there a local search method can also improve the capability of global search? To answer this question, in this work, Latin hypercube sampling with a control method is combined as the local search method.

Latin hypercube sampling (LHS) is a stratified sampling method which was proposed by Mckay in 1979 [28]. Compared with the Monte Carlo sampling, Latin hypercube sampling is more efficient and able to reflect the population more accurately from fewer sample points. Since the LHS has strong global search ability, symmetric Latin hypercube design, a variant of LHS, was used for population initialization in differential evolution [29]. Recently, LHS was successfully applied in model updating of a historic concrete bridge at two different levels [30]. On one hand, LHS was implemented with a sensitivity analysis phase to define parameters which mostly influence the FEM

modal response. It was also used to define initiation points for another optimization algorithm called Trust-Region. The application also takes advantage of its global search ability. LHS was also used as local search method for a hierarchical PSO [22]. In their work, the length of each dimension of the hypercube is two times the length of the corresponding dimension of the selected particle [22]. The setting method will make the size of the hypercube be determined by the location of the selected particle.

Actually, an optimization algorithm can be treated as a sampling method from the view of sampling. It uses some rules to a sample from the whole space of independent variables and tries to find the global optimal point. Therefore, Latin hypercube sampling can be available in CLPSO. In this work, it is used as a local search method for the global best particle in CLPSO, and this derived method is been called as CLPSO-LHS. A control method to set the size of a hypercube is also proposed in this work. At the beginning, the size of the hypercube is set as a relatively large value, which means that the hypercube for sampling is large such that the local search can improve the ability of global search. The size decreases with the iteration. In this way, the global search ability will be weakened and the local search ability will be strengthened.

Since cylindrical shapes are very common in precision components, cylindricity error evaluation problem is an important issue in assembly automation which is critical part of intelligent manufacturing. This problem belongs to minimax problems which are not differentiable and it is tough to be solved by traditional optimization methods. The least squares method is often used in most current commercial software. However, it is prone to over-estimation [31]. Recently, many metaheuristic search methods have been applied in this problem, such as improved Genetic algorithms (GA) [32], particle swarm optimization (PSO) [33], hybrid particle swarm optimization-differential evolution algorithm (PSO-DE) [31], and hierarchical PSO with Latin sampling based memetic algorithm (MA-HPSOL) [22]. The CLPSO-LHS will be applied in this problem and compared with the above-mentioned algorithms.

This paper is organized as follows: A detailed introduction of CLPSO-LHS is presented in Section 2. Then, benchmark testing for verifying the efficiency of the hybrid algorithm is shown in Section 3. Thereafter, the application of the hybrid algorithm in cylindricity error evaluation problem is presented in Section 4. And finally, Section 5 gives the conclusion of this work.

2. The hybrid algorithm CLPSO with local search based on Latin hypercube sampling (CLPSO-LHS)

2.1. PSO and comprehensive learning particle swarm optimizer (CLPSO)

In the original PSO, a particle swarm simulating a bird flock is used to search the whole space. The i^{th} particle in t^{th} iteration is denoted as $X_{i,t}=(X_{i,t}^1, X_{i,t}^1,...X_{i,t}^D)$ and its corresponding flying velocity is denoted as $V_{i,t}=(V_{i,t}^1,V_{i,t}^1,...V_{i,t}^D)$. The corresponding personal best position which memories the best position found by the i^{th} particle is denoted as $pbest_{i,t}=(pbest_{i,t}^1, pbest_{i,t}^1,...pbest_{i,t}^D)$. The global best position recording the best position discovered by the whole swarm is denoted as $gbest_t=(pbest_t^1, pbest_t^1,...pbest_t^D)$. Shi and Eberhart proposed the PSO-*w* 15 by introducing an

inertia weight (w) into the original PSO to enhance the performance. Since it improves the performance significantly, many later researchers considered PSO-w as the standard PSO. The updating equations of velocity and position in PSO-w are shown as follows:

$$V_{i.t+1} = wV_{i,t} + c_1 r_1 (pbest_{i,t+1} - X_{i,t}) + c_2 r_2 (gbest_{t+1} - X_{i,t}) \tag{1}$$

$$X_{i,t+1} = X_{i,t} + V_{i,t+1} \tag{2}$$

Where c_1 and c_2 are acceleration constants, r_1 and r_2 are random number generated by a uniform distribution in [0,1].

The comprehensive learning PSO (CLPSO) 19 adopts a learning strategy in which all other particles' historical best information is used to update a particle's velocity. The strategy makes the particles own more exemplars to learn from and a larger potential space to traverse to. The velocity updating way is shown in formula (3):

$$V_{i,t}^d = w * V_{i,t}^d + c * rand_i^d * (pbest_{fi(d),t}^d - X_{i,t}^d) \tag{3}$$

Here, $f_i = [f_i(1), f_i(2), ..., f_i(D)]$ defines which particles' *pbest*s the particle i should follow. A parameter Pc_i, referred to as the learning probability, is used to assign the value for f_i. If a random value (in the range [0, 1] in dimension d) is greater than Pc_i, i is assigned to $f_i(d)$. Otherwise, a random integer from the particles' indexes is assigned to $f_i(d)$. The Pc_i value is set by the following expression:

$$Pc_i = 0.05 + 0.45 * \frac{(\exp(10(i-1)) - 1)}{(\exp(10) - 1)} \tag{4}$$

$pbest_{fi(d)}^d$ can be the corresponding dimension of any particle's *pbest*, including its own *pbest*.

2.2. Local search based on Latin hypercube sampling

Only one sample is selected in each row or column of each sub hypercube in Latin hypercube sampling. The pseudocode of LHS is shown in Figure 1 [22]. Note that n is the dimension number of the original hypercube.

Algorithm: Latin Hypercube Sampling
Input:
Original *n*-dimensional hypercube with lower bound X_l and upper bound X_u
Sampling scale *H*
Output:
H sampling points
Step 1: For each dimension, partition the interval $[x_l^i, x_u^i]$ into *H* equivalent subintervals. As a result, H^n sub hypercubes are generated from the partition.
Step 2: Generate a sampling matrix *A*, whose scale is *H*n* and each column of which is a random permutation of [1,2,…,*H*]. Each row of the sampling matrix corresponds to a selected hypercube. As a result, *H* sub hypercubes are selected.
Step 3: Generate a sampling point from each selected hypercube. As a result, *H* sampling points are obtained.

Figure 1. Pseudocode of Latin hypercube sampling.

Figure 2 shows an example for sampling 5 points from a 2-dimensional square by LHS. It can be observed that only one sub hypercube is selected in each row or column.

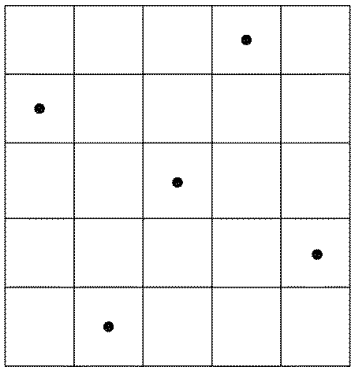

Figure 2. An example of LHS.

2.3. Control method for the size of a hypercube

When Latin hypercube sampling is used as a local search method, the size of a hypercube is a very important parameter which significantly impacts the efficiency of sampling. The parameter should decrease with the converge process. Based on the principle, the length vector of dimensions δ for a hypercube is set as formula (5). In formula (5), $\delta_{\max}$ is the initial value which is set as a ratio of (*UpB*-*LowB*). *UpB* and *LowB* denote upper boundary and lower boundary, respectively. And $\tau = Itera/\max\_\text{gen}$, where *Itera* is the current iteration number and max_gen is the maximum number of iterations. γ is a number which determines the magnitude order. p is a positive integer which is the polynomial order. k is a non-negative integer which represents the cycle for the sine function. Note that the value *Itera* is the only variable in the formula. The value δ varies with the

current iteration number *Itera*. δ and $\delta_{\max}$ are vectors in a practical problem since each dimension's *UpB* and *LowB* is different.

$$\delta = \delta_{\max} \left| 10^{-\gamma\tau^p} \sin((k+0.5)\pi(1-\tau)) \right| \tag{5}$$

Formula (5) is composed of an exponential function, a sine function and the exponential part of the exponential function is polynomial. Actually, it is obtained by trial and error. Based on the principle that δ value should decrease with the converge process, δ value was set firstly as linear control formula (6). It works well for global search. However, it converges too fast at the end leading to bad local search. Exponential control formula (7) is designed to overcome the shortage of formula (6). Further, formula (8) was designed to remaining the global search ability. At last, a sine function was added in order to improve the converge speed and search accuracy in formula (5).

$$\delta = \delta_{\max}(1-\tau) \tag{6}$$

$$\delta = \delta_{\max} \left| 10^{-\gamma\tau} \right| \tag{7}$$

$$\delta = \delta_{\max} \left| 10^{-\gamma\tau^p} \right| \tag{8}$$

When $\delta_{\max}$ is 1, γ is 10, p is 3, k is 6 and Max_gen is 5000, the plots of the four control methods are shown in Figure 3. It can be observed that the converge curve of the linear control way for formula (6) decreases to zero suddenly at the end of iteration process, which would lead to poor intensification capability, although the global search ability is strong at the beginning. The curve of the exponential control method for formula (7) decreases as a straight line. Actually, it decreases so fast at the beginning, such that it may not provide sufficient help for global search. In order to overcome this shortage, a polynomial exponential part for the exponential function is designed in formula (8). Based on formula (8), a sine function is introduced in formula (5) to improve the converge speed and search accuracy. Compared with formula (8), the δ value in formula (5) varies periodically. Since the value will reach a smaller value shortly, it can speed the convergence if the near-optimal solution has been found. At the end of iterations, the δ value can reach a smaller value, which will improve the search accuracy. Since the local search only are executed every ten iterations, τ can only get discrete values (e.g. 10/5000, 20/5000…). The value (k+0.5)(1-τ) cannot be pure integer. This is the reason why the plot of formula (5) cannot reach 0 in figure 3. In a word, formula (5) is the outcome of balance between global search ability and local search ability. These control methods will be compared in Section 3.2.

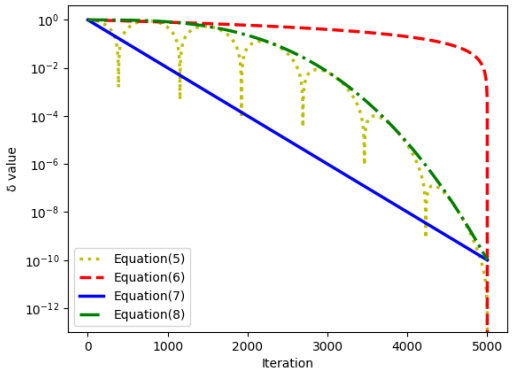

Figure 3. Plot of δ value under different control methods.

2.4. Flow chart of LHS local search

The flow chart of LHS local search is shown in Figure 4. In CLPSO-LHS, the local search is executed for the *gbest* in CLPSO for every 10 iterations. The framework of CLPSO can be found from 19.

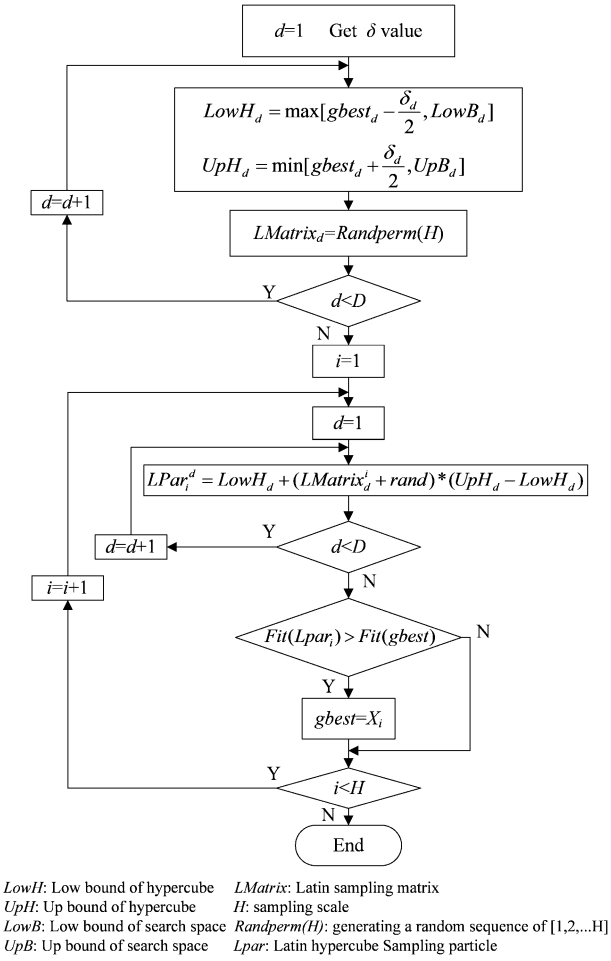

Figure 4. Flow chart of Latin hypercube sampling-based local search.

2.5. Computational complexity of CLPSO-LHS

Computational complexity analysis is important for evaluating the efficiency of an algorithm. The count of key program statements within an iteration in big O notation is used to represent computational complexity here. The CLPSO-LHS is compared with the original CLPSO. Based on the analysis from 20, CLPSO and simple PSO share the same computational complexity which is O(*ND*) where *N* is population size and *D* is dimensionality. The local search in Figure 6 includes two parts. Since the complexity of Randperm(*H*) which is used for generating a random sequence of [1,…,*H*] is O(*H*), the complexity for the first part is O(*HD*). Clearly, the complexity for the second part is also O(*HD*). The complexity for the whole local search is O(*HD*). Assuming the local search is executed for the *gbest* for every *G* iterations (*G* is 10 in current work), the total complexity for the CLPSO-LHS is O(*ND* + *HD*/*G*). Usually, *H*/*G* is less than *N* (*N* is 10 or 20, *H* and *G* is 10 in the benchmark testing), so the final complexity is still O(*ND*). As a conclusion, the computational complexity of the proposed CLPSO-LHS is the same as the original CLPSO.

3. Benchmark testing

In this section, eight benchmark functions with 10 and 30 dimensions in 19 are used to test the CLPSO with Latin hypercube sampling-based local search. The details of the test functions are attached in the Appendix.

Firstly, the proposed hybrid algorithm is compared with the original CLPSO and the pure Latin hypercube sampling-based local search in Section 3.1. The four control methods are compared in Section 3.2. The comparison with other algorithms in literature is shown in Section 3.3.

3.1. Comparison with the original CLPSO and the Latin hypercube sampling-based local search

In order to prove the efficiency of the hybrid algorithm, firstly, it was compared with the original CLPSO and the Latin hypercube sampling-based local search. Note that the results for CLPSO here are different from those in 19, since that the population sizes are different. In order to take advantage of LHS, smaller population sizes have been used here to enlarge the Max_gen. The results in 19 will be compared with CLPSO-LHS in Section 3.3.

The three algorithms were run 30 times randomly on all functions with 10 and 30 dimensions. The mean values (Mean) and standard deviation (Std) of the results were recorded. The parameters of the algorithms are shown in Table 1. CLPSO and CLPSO-LHS shared some common parameters. The initialization method in 19 was used here. For the pure LHS local search, only one solution was generated at the beginning, and formula (5) was used to set the size of hypercube. The total sampling number was set as MaxNFE value. The parameters in formula (5) were set as the same as in CLPSO-LHS except $\delta_{\max}$. The value was set as a bigger value in LHS local search such that the algorithm can search the whole solution space.

Table 1. Parameters of the algorithms.

Parameters	Value
Population size *ps*	10 for 10 and 20 for 30 dimensions
Maximum number of function evaluation MaxNFE	30,000 for 10 and 200,000 for 30 dimensions
Inertial weight *w*	$w_0 = 0.9$; $w_1 = 0.4$
Acceleration constant *c*	1.49445
Refreshing gap *m*	7
Sampling scale *H*	10
Initial length vector of dimensions for a hypercube in formula (5) $\delta_{\max}$	(*UpB*-*LowB*)/5 for CLPSO-LHS, (*UpB*-*LowB*) for pure LHS local search
Polynomial order in formula (5) *q*	3
Magnitude order in formula (5) γ	10
Cycle for the sine function in formula (5) *k*	6

Table 2. The results of the five algorithms.

No.	*D*	CLPSO Mean ± Std	LHS local search Mean ± Std	CLPSO-LHS Mean ± Std
F_1	10	1.69E-14 ± 4.33E-14	1.48E-20 ± 2.91E-21	1.37E-20 ± 9.46E-21
	30	1.69E-20 ± 1.97E-20	2.18E-21 ± 2.26E-22	6.57E-22 ± 4.01E-22
F_2	10	2.54E+00 ± 1.97E+00	8.89E+00 ± 5.36E-03	9.09E-01 ± 2.17E+00
	30	1.89E+01 ± 4.74E+00	2.87E+01 ± 7.49E-04	8.14E-01 ± 4.43E+00
F_3	10	1.41E+00 ± 3.42E+00	1.30E-10 ± 1.50E-11	1.12E-10 ± 3.01E-11
	30	1.81E+01 ± 4.49E+00	2.86E-11 ± 1.10E-12	1.70E-11 ± 5.95E-12
F_4	10	1.15E-02 ± 1.06E-02	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
	30	1.88E-13 ± 4.27E-13	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F_5	10	4.57E-10 ± 1.64E-09	1.11E-10 ± 2.73E-11	5.56E-11 ± 5.55E-11
	30	0.00E+00 ± 0.00E+00	1.69E-11 ± 1.13E-12	0.00E+00 ± 0.00E+00
F_6	10	9.63E-02 ± 2.94E-01	3.53E+00 ± 4.76E+00	0.00E+00 ± 0.00E+00
	30	8.59E-12 ± 2.05E-11	6.74E+00 ± 1.25E+01	0.00E+00 ± 0.00E+00
F_7	10	9.68E-02 ± 2.96E-01	6.77E+00 ± 4.67E+00	0.00E+00 ± 0.00E+00
	30	1.29E-01 ± 3.35E-01	2.42E+01 ± 1.19E+01	0.00E+00 ± 0.00E+00
F_8	10	6.50E+01 ± 8.42E+01	2.93E-14 ± 1.61E-13	0.00E+00 ± 0.00E+00
	30	2.67E+01 ± 4.95E+01	2.10E+02 ± 1.15E+03	0.00E+00 ± 0.00E+00

From Table 2, it can be observed that the LHS local search found good solutions for 9 out from 16 functions such as F_1, F_3, F_4 and F_5. However, for the other functions, the LHS local search performs worse than the original CLPSO. It can be inferred that the LHS local search does have strong local search ability and it also has some degree of global search ability. Otherwise, it cannot find good solutions for F_3, F_4, and F_5 which are multimodal problems. However, its global search ability is limited, so it failed for some other functions and failed on more functions with 30 dimensions which requires stronger global search ability because the solution space expands exponentially with the dimensionality. Since CLPSO has strong global search ability, it performed better than LHS local search on these functions. Since the hybrid algorithm CLPSO-LHS has the global search ability of CLPSO and the strong local search ability of LHS local search, it should

perform better than the former two algorithms. It did find best solutions for all functions. Even for F_2 which has a narrow valley from the perceived local optima to the global optimum 19, the results were much better than the original CLPSO and the pure LHS local search. For F_2 with 10 and 30 dimensions, CLPSO-LHS only failed five times and once to find the global optima, respectively. This is the reason why the standard deviation was large for F_2. In a conclusion, it is worthwhile to combine those two algorithms to form CLPSO-LHS.

3.2. Comparison of the four control methods

Here, the CLPSO with Latin hypercube sampling-based local search using formula (6) to formula (8) have been denoted as CLPSO-LHS1, CLPSO-LHS2 and CLPSO-LHS3, respectively. The testing method and parameters are the same as those in Section 3.1.

Table 3. The results of the four algorithms.

No.	*D*	CLPSO-LHS Mean ± Std	CLPSO-LHS1 Mean ± Std	CLPSO-LHS2 Mean ± Std	CLPSO-LHS3 Mean ± Std
F_1	10	1.37E-20 ± 9.46E-21	8.83E-13 ± 2.43E-12	1.68E-18 ± 6.30E-19	3.10E-18 ± 9.31E-19
	30	6.57E-22 ± 4.01E-22	2.73E-19 ± 4.28E-19	1.85E-20 ± 2.59E-20	2.09E-19 ± 2.53E-19
F_2	10	9.09E-01 ± 2.17E+00	1.91E+00 ± 3.03E+00	1.06E+00 ± 2.48E+00	1.01E+00 ± 2.65E+00
	30	8.14E-01 ± 4.43E+00	3.31E+00 ± 8.60E+00	5.04E+00 ± 1.02E+01	1.69E+00 ± 6.44E+00
F_3	10	1.12E-10 ± 3.01E-11	7.71E-03 ± 1.49E-02	1.36E-09 ± 2.35E-10	1.79E-09 ± 3.11E-10
	30	1.70E-11 ± 5.95E-12	6.61E-03 ± 8.75E-04	1.70E-09 ± 9.37E-11	1.75E-09 ± 1.45E-10
F_4	10	0.00E+00 ± 0.00E+00	4.24E-03 ± 2.68E-03	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
	30	0.00E+00 ± 0.00E+00	1.98E-12 ± 1.02E-11	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F_5	10	5.56E-11 ± 5.55E-11	4.48E-09 ± 1.05E-08	1.11E-09 ± 2.49E-09	3.28E-09 ± 5.54E-09
	30	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F_6	10	0.00E+00 ± 0.00E+00	1.10E-08 ± 3.05E-08	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
	30	0.00E+00 ± 0.00E+00	5.81E-11 ± 8.90E-11	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F_7	10	0.00E+00 ± 0.00E+00	1.02E-08 ± 5.57E-08	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
	30	0.00E+00 ± 0.00E+00	6.11E-12 ± 8.25E-12	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F_8	10	0.00E+00 ± 0.00E+00	8.21E-13 ± 2.69E-13	0.00E+00 ± 0.00E+00	0.00E-00 ± 0.00E+00
	30	0.00E+00 ± 0.00E+00	5.28E-13 ± 8.26E-13	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00

The mean values (Mean) and standard deviation (Std) of the results for the four algorithms are shown in Table 3. From the table, several items can be observed as follows. 1) Among the four algorithms, the algorithms with exponential control methods (CLPSO-LHS, CLPSO-LHS2, CLPSO-LHS3) perform better than that with the linear control method (CLPSO-LHS1). They get the global optima value (0) for 9 out 16 functions. From Figure 3, the reason can be inferred that the linear control method provides relatively large δ values in the almost whole search progress. Large δ values contribute the global search. As a result, CLPSO-LHS1 found near-optimum solutions for most functions. 2) CLPSO-LHS2 and CLPSO-LHS3 are comparable. However, CLPSO-LHS3 performs better for F_2. The difference between CLPSO-LHS2 and CLPSO-LHS3 locates at the p value in formula (8). The p value is 3 for CLPSO-LHS3 and it is 1 for CLPSO-LHS2. The p value determines the decrease speed of δ values and controls the balance between global search ability and

local search ability. Since F_2 is a hard problem which needs stronger global search ability, CLPSO-LHS3 with larger p value performs better. 3) If any other algorithms got global optimal value for a function, CLPSO-LHS also obtained its global optimal value. It performs best on those functions whose global optimal value is unable to reach. CLPSO-LHS takes formula (5) to set the δ value for LHS local search. Compared with CLPSO-LHS3, a sine function is added for two purposes. On one hand, the sine function can make the search converge in advance. On the other hand, it can provide smaller δ values which helps find more accuracy solutions. From Table 3, the second purpose was verified. The first purpose will be verified in convergence curves.

The convergence curves for the five algorithms (CLPSO is also included) for all of the functions are plotted in Figure 5 and Figure 6. Note that the iteration number is used as horizontal ordinate here. It is fair for comparison among the four CLPSO-LHS algorithms, because that they have the same number of function evaluations (NFE) at each iteration. However, the NFE in CLPSO is smaller than that in the four CLPSO-LHS algorithms. According to the parameters setting, the NFE in CLPSO approximately equals to $ps/(ps + H/10)$ times of that in CLPSO-LHS at the same iteration. Since H value is 10, the difference between the NFE and $ps/(ps + 1)$ *NFE is not big. Based on the above consideration, the iteration number is used as horizontal ordinate. By the way, since the MaxNFE is the same, the maximal iteration is larger for CLPSO.

From the two figures, it can be observed that:

1): The plots of the four hybrid algorithms are under the plots of CLSPO for most functions, especially at the beginning. Compared with CLPSO, LHS local search is added into the four hybrid algorithms. Since the local search is executed for the *gbest* particle and it has global search ability, the hybrid algorithms can find better solutions.

2): The plots of the four hybrid algorithms found good solutions for most functions, especially for CLPSO-LHS. The reason is that the LHS has good local search ability.

3): The plots of CLPSO-LHS1, CLPSO-LHS2 and CLPSO-LHS3 have similar trajectories with the plots for corresponding δ values in Figure 3 for some functions such as F_2, F_3. It can be inferred that when a near-optimum solution is found, the LHS local search works well and the search accuracy is determined by the δ value.

4): The plots of CLPSO-LHS2 converge fast for most functions except for F_2 which needs strong global search ability at the beginning. Since there is a narrow valley from the perceived local optima to the global optimum for F_2, the original CLPSO cannot find its good solution. With the help of LHS, the hybrid algorithms can find better solutions. However, if the δ value decreases too fast, the global search ability is not strong enough and the hybrid algorithm cannot find good solutions. This situation happens on CLPSO-LHS2.

5): The plots of CLPSO-LHS are staircase-like and they are under the plots of CLPSO-LHS3 for most functions which indicate that the sine function in formula (5) works well. The sine function really contributes to the convergence speed.

6): For most functions, CLPSO-LHS can reach better values at the beginning, and then, CLPSO-LHS2 exceeds it. However, CLPSO-LHS can find smaller values at the end. The sine function makes CLPSO-LHS converge fast at the beginning and it provides smaller δ values at the end leading to better final solutions.

Based on the testing results and plots, some conclusions can be made as follows: 1) The LHS local search not only contributes to the local search ability but also global search ability for CLPSO-LHS. 2) The control method shown in formula (5) can balance the global search and local search ability

well and the sine function part can accelerate the convergence and improve the final results.

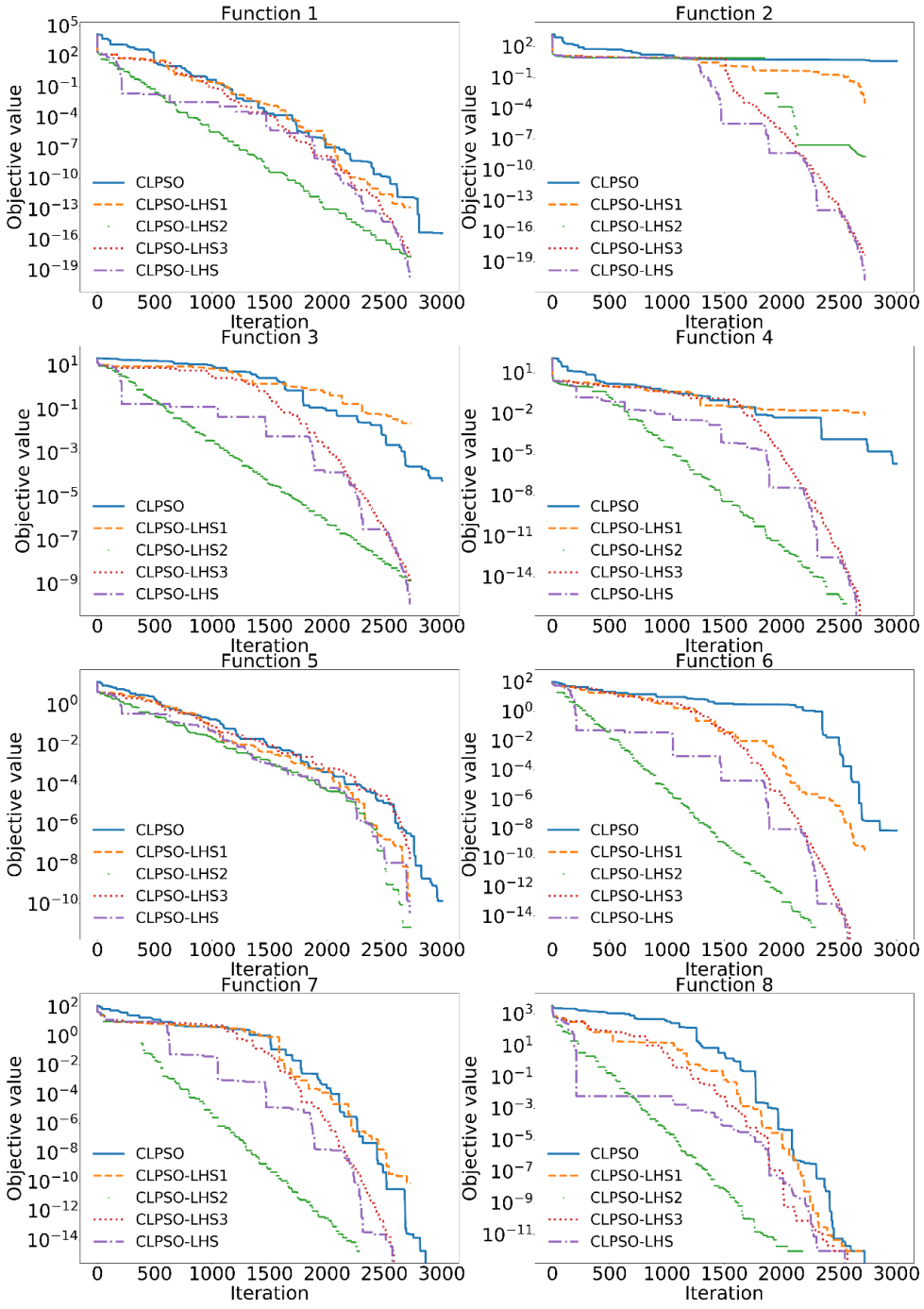

Figure 5. The convergence curves of five algorithms for 10-dimensional functions.

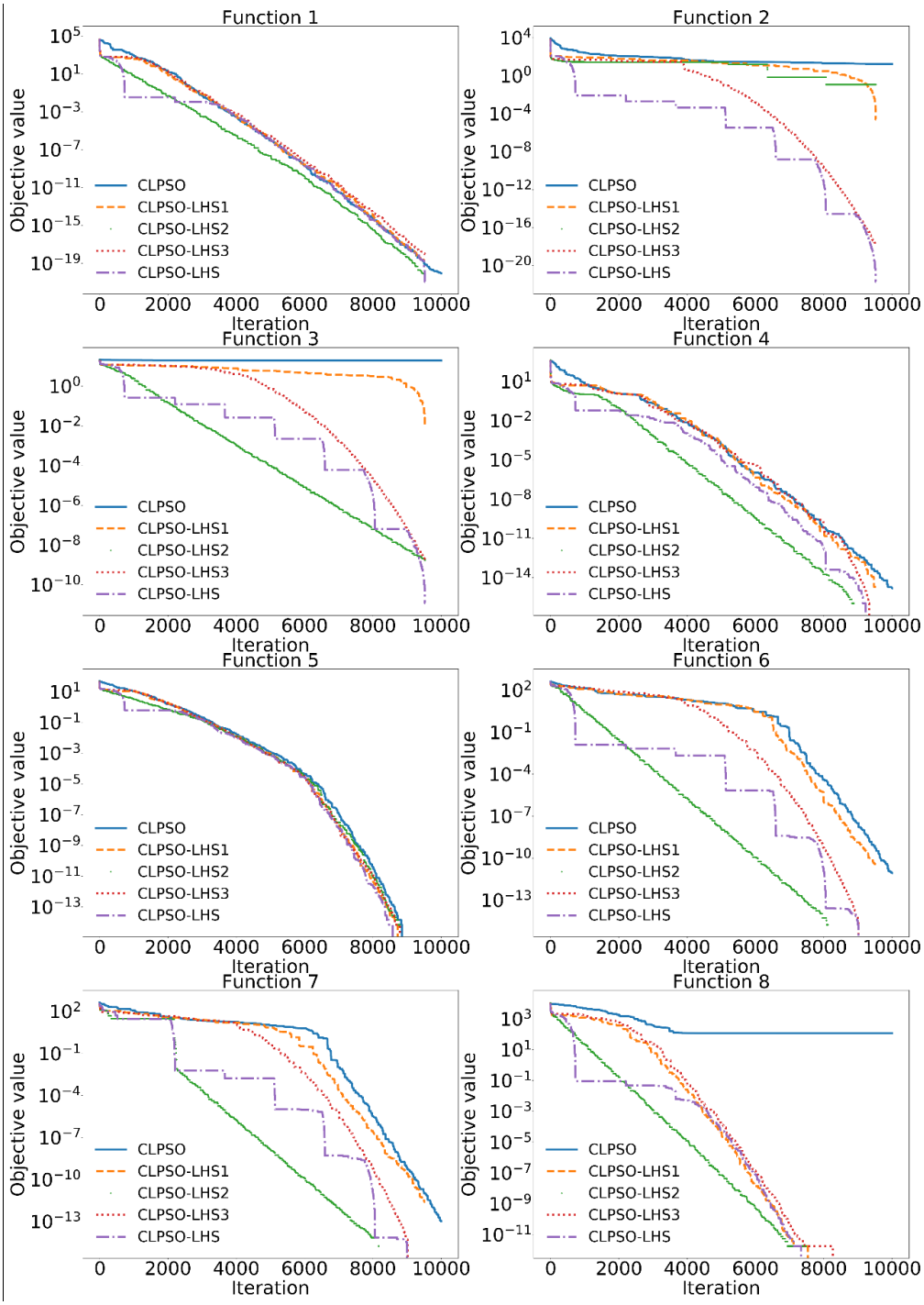

Figure 6. The convergence curves of five algorithms for 30-dimensional functions.

3.3. Comparison with other variants of PSO

In this section, CLPSO-LHS is compared with other 12 variants of PSO from literature. The 12 algorithms are listed as follows.

- PSO with inertia weight (PSO-*w*) 15
- PSO with constriction factor (PSO-cf) [34]
- Local version of PSO with inertia weight (PSO-*w*-local) 19

- Local version of PSO with constriction factor (PSO-cf-local) 16
- UPSO 18
- Fully informed particle swarm (FIPS) 17
- FDR-PSO [35]
- CPSO-H [36]
- CLPSO 19
- CPSO-inner 20
- CPSO-outer 20
- PCLPSO 23

The first nine algorithms' results are obtained from 19. For 10-dimentional functions, the nine algorithms' population size is 10 and the maximum number of function evaluation is 30,000. For 30-dimentional functions, the population size is 40 and the maximum number of function evaluation (MaxNFE) is 200,000. CPSO-inner and CPSO-outer hybridized cellular automata in PSO. The results are from 20. The CPSO-inner is the hexagonal version which performed best among the three versions. In the two versions of CPSO, the population size is 36 and the maximum number of iterations is set as 5,000 for both 10-dimensional and 30-dimensional functions. The MaxNFE in CPSO-inner is 180,000 ($36 \times 5{,}000$) and the value is 230,000 ($(36 + 10) \times 5{,}000$) in CPSO-outer. PCLPSO 23 is a better variation of CLPSO. It uses a novel parallel multi-swarm strategy. The setting of population size and MaxNFE is the same as in the first nine algorithms. The parameters for CLPSO-LHS are set as in Table 1. The independent running number is 30 for all functions.

The results of all the algorithms on 10-dimensional and 30-dimensional functions are shown in Table 4 and Table 5. The best results for each function are stressed by bold.

From Table 4, 5, it can be observed that CLPSO-LHS performed best for 10 out from 16 functions. CPSO-outer had best results for 7 10-dimensional functions since its MaxNFE was much larger. PCLSPO performed best for 6 functions. CLPSO archived 5 best results. CPSO-inner archived best results on 3 functions and CPSO-H performs best on 2 functions. Benefiting from the LHS local search, CLPSO-LHS still was able to achieve good results when almost all the other algorithms failed for F_2. Only CLPSO, PCLPSO and CLPSO-LHS, three CLPSO based algorithms, archived good results for F_8. For 10-dimensional F_4, only CLPSO-LHS and CPSO-outer obtained good results. CLPSO-LHS still obtained good results for the other six functions where it didn't perform best. PCLPSO and CLPSO performed very well on most functions. CPSO-outer shows good performance in 10-dimensional functions with the larger MaxNFE, but it is unable to perform well on most 30-dimensional functions. From an overall perspective, among all the compared algorithms, CLPSO-LHS outperformed other algorithms, especially for 30-dimentional functions.

Table 4. The results for some variants of PSO on the 10-dimensional functions.

Algorithm		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
PSO-w	Mean	7.96E-51	3.08E+00	1.58E-14	9.69E-02	2.28E-03	5.82E+00	4.05E+00	3.20E+02
	Std	3.56E-50	7.69E+01	1.60E-14	5.01E-02	7.04E-03	2.96E+00	2.58E+00	1.85E+02
PSO-cf	Mean	9.84E-105	6.98E+01	9.18E-01	1.19E-01	6.69E-01	1.25E+00	1.20E+01	9.87E+02
	Std	4.21E-104	1.46E+00	1.01E+00	7.11E-02	7.17E-01	5.17E+00	4.99E+00	2.76E+02

Continued on next page

Algorithm		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
PSO-w-local	Mean	2.13E-35	3.92E+00	6.04E-15	7.80E-02	1.41E-06	3.88E+00	4.77E+00	3.26E+02
	Std	6.17E-35	1.19E+00	1.67E-15	3.79E-02	6.31E-06	2.30E+00	2.84E+00	1.32E+02
PSO-cf-local	Mean	1.37E-79	8.60E+01	5.78E-02	2.80E-02	7.85E-02	9.05E+00	5.95E+00	8.78E+02
	Std	5.60E-79	1.56E+00	2.58E-01	6.34E-02	5.16E-02	3.48E+00	2.60E+00	2.93E+02
UPSO	Mean	9.84E-118	1.40E+00	1.33E+00	1.04E-01	1.14E+00	1.17E+00	5.85E+00	1.08E+03
	Std	3.56E-117	1.88E+00	1.48E+00	7.10E-02	1.17E+00	6.11E+00	3.15E+00	2.68E+02
FDR	Mean	2.21E-90	8.67E-01	3.18E-14	9.24E-02	3.01E-03	7.51E+00	3.35E+00	8.51E+02
	Std	9.88E-90	1.63E+00	6.40E-14	5.61E-02	7.20E-03	3.05E+00	2.01E+00	2.76E+02
FIPS	Mean	3.15E-30	2.78e+00	3.75E-15	1.31E-01	2.02E-03	2.12E+00	4.35E+00	7.10E+01
	Std	4.56E-30	2.26E-01	2.13E-14	9.32E-02	6.40E-03	1.33E+00	2.80E+00	1.50E+02
CPSO-H	Mean	4.98E-45	1.53E+00	1.49E-14	4.07E-02	1.07E-15	0.00E+00	2.00E-01	2.13E+02
	Std	1.00E-44	1.70E+00	6.97E-15	2.80E-02	1.67E-15	0.00E+00	4.10E-01	1.41E+02
CLPSO	Mean	5.15E-29	2.46E+00	4.32E-14	4.56E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	2.16E-28	1.70E+00	2.55E-14	4.81E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
CPSO-inner3	Mean	1.26E-03	6.89E+00	0.00E+00	6.83E-02	0.00E+00	1.22E-01	0.00E+00	1.94E+03
	Std	2.32E-03	1.14E+00	0.00E+00	3.34E-02	0.00E+00	2.85E-01	0.00E+00	4.25E+02
CPSO-outer	Mean	0.00E+00	2.66E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.21E+02
	Std	0.00E+00	1.01E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.17E+02
PCLPSO	Mean	4.29E-57	5.84E-01	4.00E-15	6.59E-04	0.00E+00	0.00E+00	0.00E+00	1.27E-04
	Std	1.17E-56	1.59E+00	2.41E-30	2.50E-03	0.00E+00	0.00E+00	0.00E+00	2.78E-13
CLPSO-LHS	Mean	1.37E-20	9.09E-01	1.12E-10	0.00E+00	5.56E-11	0.00E+00	0.00E+00	0.00E+00
	Std	9.46E-21	2.17E+00	3.01E-11	0.00E+00	5.55E-11	0.00E+00	0.00E+00	0.00E+00

Table 5. The results for some variants of PSO on the 30-dimensional functions.

Algorithm		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
PSO-w	Mean	9.78E-30	2.93E+01	3.94E-14	8.13E-03	1.30E-04	2.90E+01	2.97E+01	1.10E+03
	Std	2.50E-29	2.51E+01	1.12E+00	7.16E-03	3.30E-04	7.70E+00	1.39E+01	2.56E+02
PSO-cf	Mean	5.88E-100	1.11E+01	1.12E+00	2.06E-02	4.10E+00	5.62E+01	2.85E+01	3.78E+03
	Std	5.40E-100	1.81E+00	8.65E-01	1.90E-02	2.20E+00	9.76E+00	1.14E+01	6.02E+02
PSO-w-local	Mean	5.35E-100	2.39E+01	9.10E-08	5.91E-03	4.94E-03	2.72E+01	2.08E+01	1.53E+03
	Std	4.41E-13	3.07E+00	8.11E-08	6.69E-03	1.40E-02	7.58E+00	4.94E+00	3.00E+02
PSO-cf-local	Mean	7.70E-54	1.71E+01	5.33E-15	5.91E-03	1.16E-01	4.53E+01	1.54E+01	3.78E+03
	Std	1.59E-53	9.16E-01	1.87E-15	8.70E-03	2.79E-01	1.17E+01	1.67E+01	5.37E+02
UPSO	Mean	4.17E-87	1.51E+01	1.22E-15	1.66E-03	9.60E+00	6.59E+01	6.34E+01	4.84E+03
	Std	3.15E-87	8.14E-01	3.16E-15	3.07E-03	3.78E+00	1.22E+01	1.24E+01	4.76E+02
FDR	Mean	4.88E-102	5.39E+00	2.84E-14	1.01E-02	7.49E-03	2.84E+01	1.44E+01	3.61E+03
	Std	1.53E-101	1.76E+00	4.10E-15	1.23E-02	1.14E-02	8.71E+00	6.28E+00	3.06E+02
FIPS	Mean	2.69E-12	2.45E+01	4.81E-07	1.16E-06	1.54E-01	7.30E+01	6.08E+01	2.05E+03
	Std	6.84E-13	2.19E-01	9.17E-08	1.87E-06	1.48E-01	1.24E+01	8.35E+00	9.58E+02
CPSO-H	Mean	1.2E-113	7.08E+00	4.93E-14	3.63E-02	7.82E-15	0.00E+00	1.00E-01	1.08E+03
	Std	2.9E-113	8.01E+00	1.10E-14	3.60E-02	8.50E-15	0.00E+00	3.16E-01	2.59E+02
CLPSO	Mean	4.46E-14	2.10E+01	0.00E+00	3.14E-10	3.45E-07	4.85E-10	4.36E-10	1.27E-12
	Std	1.73E-14	2.98E+00	0.00E+00	4.64E-10	1.94E-07	3.63E-10	2.44E-10	8.79E-13

Continued on next page

Algorithm		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
CPSO-inner3	Mean	1.60E+00	2.90E+01	3.13E+00	1.50E-01	1.15E+00	1.38E+00	1.42E-01	7.60E+03
	Std	6.94E-01	6.57E-01	6.79E-01	5.87E-02	1.85E+00	4.75E+00	3.46E-01	1.49E+03
CPSO-outer	Mean	9.48E-71	1.01E+00	5.03E-15	1.52E-02	1.91E+00	5.00E+01	7.52E+01	1.51E+03
	Std	5.13E-70	6.65E-01	2.90E-15	2.22E-02	1.38E+00	2.49E+01	3.52E+01	1.42E+03
PCLPSO	Mean	2.87E-28	6.63E+00	2.05E-14	9.22E-12	0.00E+00	0.00E+00	0.00E+00	3.82E-04
	Std	6.84E-28	1.17E+01	6.29E-15	4.25E-11	0.00E+00	0.00E+00	0.00E+00	7.40E-13
CLPSO-LHS	Mean	6.57E-22	8.14E-01	1.70E-11	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Std	4.01E-22	4.43E+00	5.95E-12	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

4. Cylindricity error evaluation problem

The calculation of cylindricity error is illustrated in Figure 7. Assuming that the axis direction is n(m, n, 1) and the radius is R. The aim of the problem is to find an optimal axis direction n which passes the point Q(x_0, y_0, 1) to minimize the cylindricity error. The variants are m, n, x_0 and y_0. Given a point set P = {P_k|k = 1, 2,…, M}, assuming that the length from point P_1 to the axis is longest and the length from P_2 to the axis is shortest, the cylindricity error is,

$$f = |F_1P_1| - |F_2P_2| = (|E_1P_1| - R) - (|E_2P_2| - R) = |E_1P_1| - |E_2P_2| \tag{7}$$

If both $|E_1P_1|$ and $|E_2P_2|$ are bigger than R. When $|E_1P_1|$ is bigger than R and $|E_2P_2|$ is shorter than R, the cylindricity error is,

$$f = |F_1P_1| + |F_2P_2| = (|E_1P_1| - R) + (R - |E_2P_2|) = |E_1P_1| - |E_2P_2| \tag{8}$$

When both $|E_1P_1|$ and $|E_2P_2|$ are shorter than R, the same formula is obtained. So the objective function is written as formula (9).

$$f = |EP_1| - |EP_2| = \max_k \frac{|\vec{QP_k} \times \mathbf{n}|}{|\mathbf{n}|} - \min_k \frac{|\vec{QP_k} \times \mathbf{n}|}{|\mathbf{n}|} \tag{9}$$

Where

$$\frac{|\vec{QP_k} \times \mathbf{n}|}{|\mathbf{n}|} = \frac{\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ x_k - x_0 & y_k - y_0 & z_k \\ l & m & 1 \end{vmatrix}}{\sqrt{l^2 + m^2 + 1}} \tag{10}$$

Four variables (x_0, y_0, l, m) were initialized in [−1, 1]. Two data sets from [33] and [37] were used to test the hybrid algorithm. The part for the first data set is with the dimension of 39 mm in radius and 120 mm in length. The measurement data for the cylindrical outer surface contains 80 records. The radius and the length of the part for the second data set is 60 mm and 30 mm, respectively. There are 20 data records. The data records are formatted as (x_k, y_k, z_k) which is the

input for the objective function as formula (9).

The population size of the CLPSO-LHS was set to be 6 and the MaxNFE was set to be 40000. The other parameters were the same as those in Section 3.2. The algorithms were run 30 times independently on the two data sets. The statistical results (best, worst, median, mean value and the standard deviation) are shown as in Table 6.

Table 6. The statistical results of the CLPSO-LHS for cylindricity error evaluation problems.

	Best	Worst	Median	Mean	Std
Data set 1	1.04864E-02	1.09565E-02	1.04864E-02	1.05072 E-02	8.38472E-05
Data set 2	1.65742E-01	1.73070E-01	1.67500E-01	1.67889E-01	1.75039E-03

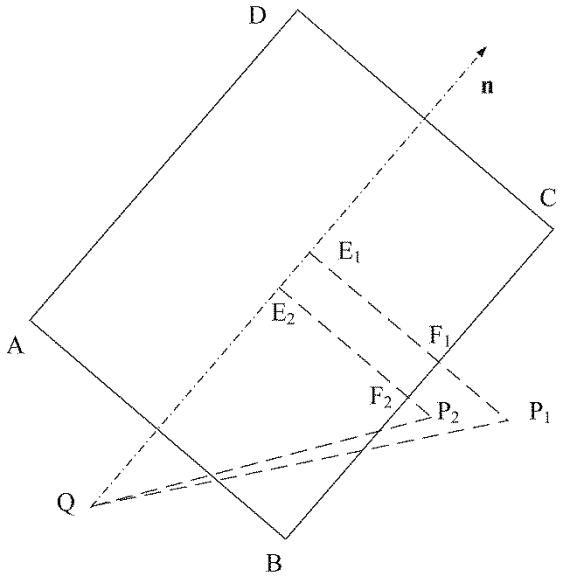

Figure 7. The illustration of calculating the cylindricity error.

Table 7 shows the comparison of best solutions between the CLPSO-LHS and other algorithms. The compared algorithms include improved GA1 [32], improved GA2 [38], PSO [33], MA-HPSOL [22], and PSO-DE [31].

Table 7. The comparison of best solutions between CLPSO-LHS and other algorithms for cylindricity error evaluation problems.

Parameter	Data set 1				Data set 2			
	Improved GA1	PSO	MA-HPSOL	CLPSO-LHS	Improved GA2	PSO-DE	MA-HPSOL	CLPSO-LHS
x_0	0.0009250	0.003315	0.0020284	0.00181322	0.011853	0.010650	0.0106429	−0.0105382
y_0	−0.0002253	0.002814	0.0000496	4.67743e-5	0.047689	0.046918	0.0469181	0.0909596
l	0.0000435	−0.00052	0.0000591	4.04684e-5	−0.000674	−0.000619	−0.000619	0.00121953
M	0.0000162	0.000609	0.0000214	1.52445e-5	0.002960	−0.002915	−0.002915	−0.006714
Cylindricity	0.0105976	0.025368	0.0104864	0.0104864	0.184274	0.183957	0.1839592	0.165742

CLPSO-LHS and MA-HPSOL perform best for the first data set. The best solution obtained by them is much better than the one obtained by PSO. For the second data set, CLPSO-LHS is the best one and it is much better than others including MA-HPSOL.

5. Conclusions

In this paper, a local search based on Latin hypercube sampling is combined with CLPSO. The local search shows a good performance when a control method is used to control the size of the hypercube. The numerical experiments show that, at the beginning, the local search method contributes to global search ability. Ultimately, it provides good fine search ability. The algorithm performs well on testing functions. The hybrid algorithm also shows a good performance on its application in cylindricity error evaluation problems.

Although the algorithm is very effective, there is still some room for improvement. First, the designed control method has lots of parameters. A self-adaptive control method can be developed and its parameters can be reduced in the future. Second, although the local search method helps to find a good solution at the beginning, it can be observed from convergence figures that the solution cannot be improved for a short time for some functions, such as F_1, F_5 and F_8. How to utilize the global search ability to accelerate the converge speed deserves further study.

Acknowledgement

This research work is supported in part by National Natural Science Foundation of China (NSFC) under Grant No. 61603145, in part by Natural Science Foundation of Hubei Province under Grant No. 2015CFB528, and in part by Open Project of State Key Lab of Digital Manufacturing Equipment & Technology under Grant DMETKF2018011.

Conflicts of interest

All authors declare no conflict of interest in this paper.

Reference

1. S. Krikpatrick, C. D. Gelatt and M. Vecchi, Optimization by simulated annealing, *Science.*, **220** (1983), 671–680.
2. J. H. Holland, Genetic algorithms and the optimal allocation of trials, S*iam. J. Comput.*, **2** (1973), 88–105.
3. R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, In: *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on* IEEE, 39–43.
4. R. Storn and K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Global. Optim.*, **11** (1997), 341–359.
5. Y. Z. Zhou, W. C. Yi and L. Gao, et al., Adaptive differential evolution with sorting crossover rate for continuous optimization problems, *IEEE T. Cy.*, **47** (2017), 2742–2753.

6. M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization: artificial ants as a computational intelligence technique, *IEEE. Comput. Intell. Mag.*, **1** (2006), 28–39.
7. S. Birbil and S. C. Fang, An electromagnetism-like mechanism for global optimization, *J. Global. Optim.*, **25** (2003), 263–282.
8. W. Gong, Z. Cai and D. Liang, Engineering optimization by means of an improved constrained differential evolution, *Comput. Method. Appl. M.*, **268** (2014), 884–904.
9. Q. Wu, L. Gao and X. Y. Li, et al., Applying an electromagnetism-like mechanism algorithm on parameters optimization of a multi-pass milling process, *Int. J. Prod. Res.*, **51** (2012), 1777–1788.
10. C. Zhang, Q. Lin and L. Gao, et al., Backtracking Search Algorithm with three constraint handling methods for constrained optimization problems, *Expert. Syst. Appl.*, **42** (2015), 7831–7845.
11. X. Y. Li, L. Gao and Q. K. Pan, et al., An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop, *IEEE. T. Syst. Man. Cy. A.*
12. X. Y. Li, C. Lu and L. Gao, et al., An Effective Multi-Objective Algorithm for Energy Efficient Scheduling in a Real-Life Welding Shop, *IEEE T. Ind. Inform.*, **14** (2018), 5400–5409.
13. X. Y. Li and L. Gao, An Effective Hybrid Genetic Algorithm and Tabu Search for Flexible Job Shop Scheduling Problem, *Int. J. Prod. Econ.*, **174** (2016), 93–110.
14. V. Vassiliadis and G. Dounias, Nature–inspired intelligence: a review of selected methods and applications. *Int. J. Artif. Intell. T.*, **18** (2009), 487–516.
15. Shi Y, Eberhart R, A modified particle swarm optimizer, In: *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence.*, 69–73.
16. J. Kennedy and R. Mendes, Population structure and particle swarm performance, In: *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on IEEE*, **2** (2002) 1671–1676.
17. R. Mendes, J. Kennedy and J. Neves, The fully informed particle swarm: Simpler, maybe better, *IEEE T. Evolut. Comput.*, **8** (2004), 204–210.
18. Parsopoulos K E, Vrahatis M N, UPSO – a unified particle swarm optimization scheme, *Lecture Series on Computational Sciences*, **1** (2004), 868–873.
19. J. J. Liang, A. K. Qin and P. N. Suganthan, et al., Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE T. Evolut. Comput.*, **10** (2006), 281–295.
20. Y. Shi, H. Liu and L. Gao, et al., Cellular particle swarm optimization, *Inform. Sci.*, **181** (2011), 4460–4493.
21. W. H. Lim and N. A. M. Isa., Adaptive division of labor particle swarm optimization, *Expert. Syst. Appl.*, **42** (2015), 5887–5903.
22. Y. Peng and B. Lu, A hierarchical particle swarm optimizer with latin sampling based memetic algorithm for numerical optimization, *Appl. Soft. Comput.*, **13** (2013), 2823–2836.
23. Ş. Gülcü and H. Kodaz, A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, *Eng. Appl. Artif. Intel.*, **45** (2015), 33–45.
24. Y. Bao, Z. Hu and T. Xiong, A PSO and pattern search based memetic algorithm for SVMs parameters optimization, *Neurocomputing*, **117** (2013), 98–106.
25. Y. G. Petalas, K. E. Parsopoulos and M. N. Vrahatis, Memetic particle swarm optimization, *Ann. Oper. Res.*, **156** (2007), 99–127.

26. D. Jia, G. Zheng and B. Qu, et al., A hybrid particle swarm optimization algorithm for high-dimensional problems, *Comput. Ind. Eng.*, **61** (2011), 1117–1122.
27. G. Wu, D. Qiu and Y. Yu, et al., Superior solution guided particle swarm optimization combined with local search techniques, *Expert. Syst. Appl.*, **41** (2014), 7536–7548.
28. M. D. McKay, R. J. Beckman and W. J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, **21** (1979), 239–245.
29. Z. Zhao, J. Yang and Z. Hu, et al., A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems, *Eur. J. Oper. Res.*, **250** (2016), 30–45.
30. R. Ferrari, D. Froio and E. Rizzi, et al., Model updating of a historic concrete bridge by sensitivity- and global optimization-based Latin Hypercube Sampling, *Eng. Struct.*, **179** (2019), 139–160.
31. X. Zhang, X. Jiang and P. J. Scott, A reliable method of minimum zone evaluation of cylindricity and conicity from coordinate measurement data, *Precis. Eng.*, **35** (2011), 484–489.
32. H. Lin and Y. Peng, Evaluation of cylindricity error based on an improved GA with uniform initial population, In: *Control, Automation and Systems Engineering, 2009. CASE 2009. IITA International Conference on IEEE*, 311–314.
33. J. Mao, Y. Cao and J. Yang, Implementation uncertainty evaluation of cylindricity errors based on geometrical product specification (GPS), *Measurement*, **42** (2009), 742–747.
34. M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE T. Evolut. Comput.*, **6** (2002), 58–73.
35. T. Peram, K. Veeramachaneni and C. K. Mohan, Fitness-distance-ratio based particle swarm optimization, In: Proceedings of Swarm Intelligence Symposium, (2003), 174–181.
36. F. Van den Bergh and A. P. Engelbrecht, A cooperative approach to particle swarm optimization, *IEEE T. Evolut. Comput.*, **8** (2004), 225–239.
37. K. Carr and P. Ferreira, Verification of form tolerances part II: Cylindricity and straightness of a median line, *Precis. Eng.*, **17** (1995), 144–156.
38. X. Wen and A. Song, An improved genetic algorithm for planar and spatial straightness error evaluation, *Int. J. Mach. Tool. Manu.*, **43** (2003), 1157–1162.

AIMS Press